DeepGeo: Photo Localization with Deep Neural Network

Montiel Abello
mabello@andrew.cmu.edu

Nathaniel Chodosh
nchodosh@andrew.cmu.edu

Sudharshan Suresh
sudhars1@andrew.cmu.edu

1 Introduction

Games such as GeoGuessr demonstrate that humans are remarkably good at coarse estimates of location from a single image. The game presents the player with a Google Street View $360°$ panorama from anywhere in the world and asks the user to guess the GPS coordinates of the place the photo was captured. We are able to achieve decent ballpark estimates in spite of the fact that many (if not most) images have ambiguous locations unless they contain very specific landmarks.

Natural images have numerous cues that allow inference, such as vegetation, and man-made infrastructure such as roads, marking and signs. Humans are clearly able to use additional rough cues to find out where they are. Common examples of these are driving directions, languages and prior knowledge of vegetation and geography. In this project we will attempt to replicate this semantic reasoning using a deep neural network. The abundance of open-source visual data and previous success of learning methods in scene recognition motivates this effort.

Previous works, based on both learning as well as scene matching methods, have shown appreciable results at various spatial scales. DeepGeo targets a classification problem, that spans a restricted scope - the United States. The United States is an ideal playground for such a method due to its near-perfect street view coverage, high road density and varied geography. Automatic image localization is an interesting and challenging problem that will tests the capabilities of deep neural networks. This tool will also aid in wide-ranging applications such as meta-data aggregation, location-based priming and the creation of richer datasets.

Our key contributions are **(a)** a deep neural network classifier that predicts the most-probable state a given image was captured in (not restricted to cities/landmarks) **(b)** the `50State10K` dataset, a comprehensive street view dataset of the Unites States.

More concretely, given a set of input samples $X = \{X_1, X_2, \ldots, X_n\}$ where $X_i \in \mathbb{R}^{4\times256\times256}$, and a set of one-hot vector labels $Y = \{Y_1, Y_2, \ldots, Y_n\}$ where $Y_i \in \mathbb{R}^{50}$, we create a model to predict the distribution over all states that a test sample X' consists of images belonging to class $P(Y' = Y_i|X')$.

2 Related Work

The challenges of geolocation based on visual information has its foundations in [1]. There have been various formulations of the same problem over the years that incorporate geotagged online photos [2, 3, 4], satellite aerial imagery [5] and city skylines [6]. Due to their extensive coverage and accuracy, Google Street View scenes have been used in [7, 8, 9]. These works however only tackle localization at the city level, most using feature-based methods.

PlaNet [3] and IM2GPS [2] are considered the baselines for scalable image geolocation over the entire world. IM2GPS computes the closest match via scene matching with a large corpus of geotagged Flickr images. PlaNet [3] uses a larger such dataset paired with a deep network to obtain a probability distribution of location over a partitioned world map. These methods rely on sheer volume of training images - a perusal of their data yields many indoor scenes, pictures of food and pets.

They are also naturally primed towards famous landmarks and cities, and are not representative of *throw a dart at a map* scenarios such as GeoGuessr. As mentioned in section 1, our method differs from the aforementioned works both in terms of classification goals and data collection. While PlaNet correctly classifies 28.4% of the test data at the country level, IM2GPS has a test set median error of 500km. It is worth mentioning that PlaNet bested human subjects in GeoGuessr in 28 rounds of 50.

3 Methods

3.1 Location Sampling

To sample locations within each state, we use state boundary data from [10]. State boundaries are represented by a polygon connecting an ordered set of points. We sample `[latitude, longitude]` uniformly within the minimum rectangle enclosing this polygon. Points falling outside the state boundary are discarded. The *Gridded Population of the World v4* [11] dataset is used to inform our location sampling. This density-based sampling allows us to both achieve a good spread of locations as well as priming the model for more densely populated regions.

Consequently, it also boosts the hit rate of our image scraper described in section 3.2. We use the population density map with a grid size of 2.5 arc-minutes in latitude and longitude. Within each state we normalize the population density values by dividing each grid value by the maximum density in the state, and discard sampled locations with a population density below a threshold p_{min}.

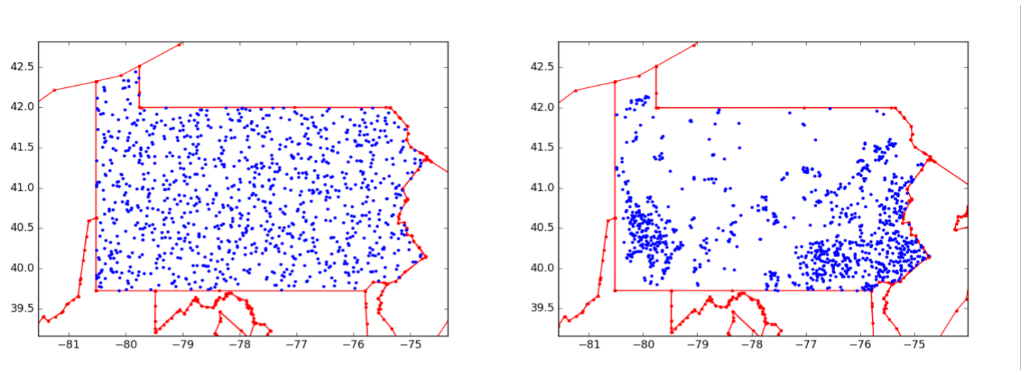

Figure 1: Sample locations in Pennsylvania for uniform and our population density aware approaches.

3.2 Data Scraping

Out dataset is obtained via the Google Street View API, which provides tools for image extraction. The sampled locations from section 3.1 are fed as queries to a script[1], which retrieves the closest unique panorama. These 360° images are tiled and split to the 4 cardinal directions. Duplicate images are pruned away to give `4 x 2500 x 50` unique natural scene images. The dataset is described in further detail in section 4.

3.3 Baseline Training Method

Before attempting any novel approaches to this problem we first decided to try a naive method in order to have a baseline to compare with. Since this is essentially an image classification problem we decided to train a standard deep CNN on the task. The Residual Network (ResNet) is a recently published CNN architecture which we determined to be suitable for our problem [12]. The ResNet architecture uses so called *residual blocks* to achieve better gradient propagation, which enables training much deeper networks. In [12] the authors present variations of their network that have 18 to over 100 layers, while we selected the 18 and 50 layer variants for our tests to get an idea of how helpful network capacity is in our task. We implemented the network in Tensorflow as described in [12] and used the Adam algorithm for training with default parameters.

4 Dataset `[50State10K]`

4.1 Training

Our dataset consists of 2500 samples for each of the 50 states for a total of 125000 samples x_i. Each sample contains the geographical coordinates of the sample location $< \texttt{lat}_i, \texttt{long}_i >$, state

[1]based on Adrian Letchford's python module.

label $y_i \in [0, 49]$ and four 256×256 Street View images with an elevation of $0°$ and headings of $\{0°, 90°, 180°, 270°\}$ respectively, stored with a *.jpg* file format. This gives us a total of $125000 \times 4 = 500000$ unique images across the country. To the best of our knowledge, this is the most comprehensive such dataset of the United States.

Samples were obtained by uniformly sampling geographical coordinates with a normalized population density greater than $p_{min} = 10^{-2}$ within each state. We use the individual samples labeled with a cardinal direction $h_i \in \{0, 1, 2, 3\}$ for the training the network. Figure 2 shows example images from two such samples.

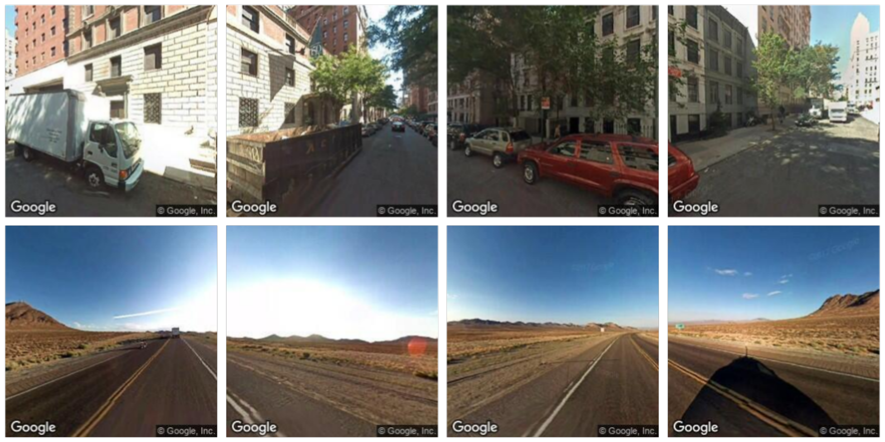

Figure 2: Two input samples consisting of four images each at headings of $[0°, 90°, 180°, 270°]$.

4.2 Testing

We generate an independent set of test images, using a similar procedure to dataset generation. To ensure a good spread of candidates across each state, we define a heuristic so that the sampled geographical coordinates are separated by a minimum distance (e.g. 100m). We generate a set that accounts for 20% of `[50State10K]`.

An additional test metric, inspired by [3], compares the network's performance against a human. The network predicts states of a random location generated in GeoGuessr. The responses of the network are compared with that of a human subject and the ground truth.

5 Preliminary Results

Since collecting a large dataset on all 50 states is quite time consuming, we started by running our baseline method on two smaller datasets which cover 5 states each. For each dataset we also used 10% hold-out set for testing. The first dataset consists of the states: California, Hawaii, North Carolina, North Dakota, and Pennsylvania (Figure 3). We chose these states to be easy to discriminate since they all have relatively different geographical conditions. The baseline 18 layer method performed reasonably well on this dataset, achieving **70%** accuracy. We then created a second dataset, this time alphabetically. The second dataset consists of: Alabama, Arizona, Arkansas, California, and Colorado. As expected the performance dropped on this dataset to 58% for the 18 layer variant of our network. We then tried the 50 layer version which achieved **63%** accuracy[2].

[2]Bear in mind random chance for 50 states is 2%

Figure 3: 100 images - California, Hawaii, North Carolina, North Dakota, and Pennsylvania datasets

Based on the initial results from 5 states, we believe unsupervised clustering of our test data will not be necessary, as we previously thought. We have instead decided to experiment with several approaches for incorporating the image heading information in training our network. We have two initial ideas for incorporating this information:

- **Early Integration**: Combine all four images (NSEW for each location) into a 12 channel input and train the network as before. We expect this method to be most helpful if there is important information encoded in the direction itself. e.g: seeing the ocean in the west image probably means it was taken on the west coast.
- **Late Integration**: Run the same network on all 4 images and then maxpool the predictions. We expect this method to be better if there isn't much useful information in the actual direction but using all 4 headings helps mitigate getting a "bad view" in one direction. We also intend to experiment with different integration schemes, such as concatenating the final convolution features from all 4 images and then performing the the classification on those.

6 Timeline and Work Division

The table shows our project timeline and progress, as well as responsible team members for future tasks. We were able to fulfill our goals for the midterm report and at the time of writing we have collected the full dataset and are preparing our network architecture for initial training.

Date	**Task**	**Responsible**
March 30	Google Street View API interface, generate small dataset	Sudharshan
April 2	State Borders and Population Density	Montiel
April 6	Initial results from toy example	Nathaniel
April 11	Milestone report due	All
April 13	Generate full dataset	Montiel, Sudharshan
April 15	Incorporate heading labels for training data	Montiel, Sudharshan
April 18	Initial results on full dataset	Nathaniel
April 20	Analyse results and make changes to network architecture.	All
April 23	Collect final results	Nathaniel
April 27	Write initial paper draft and complete poster.	All
April 30	Poster presentation	All
May 1	Finalize paper.	All
May 2	Final report due	All

References

[1] William B Thompson, Carolyn M Valiquette, BH Bennet, and Karen T Sutherland. Geometric reasoning for map-based localization. *Computer Science Technical Report UUCS-96-006, University of Utah, Salt Lake City, UT*, 1996.

[2] James Hays and Alexei A Efros. Im2gps: estimating geographic information from a single image. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[3] Tobias Weyand, Ilya Kostrikov, and James Philbin. Planet-photo geolocation with convolutional neural networks. In *European Conference on Computer Vision*, pages 37–55. Springer, 2016.

[4] Song Cao and Noah Snavely. Graph-based discriminative learning for location recognition. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 700–707. IEEE, 2013.

[5] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5307, 2016.

[6] Srikumar Ramalingam, Sofien Bouaziz, Peter Sturm, and Matthew Brand. Skyline2gps: Localization in urban canyons using omni-skylines. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3816–3823. IEEE, 2010.

[7] Amir Roshan Zamir and Mubarak Shah. Accurate image localization based on google maps street view. In *European Conference on Computer Vision*, pages 255–268. Springer, 2010.

[8] Amir Roshan Zamir and Mubarak Shah. Image geo-localization based on multiplenearest neighbor feature matching usinggeneralized graphs. *IEEE transactions on pattern analysis and machine intelligence*, 36(8):1546–1558, 2014.

[9] Hyo Jin Kim, Enrique Dunn, and Jan-Michael Frahm. Predicting good features for image geo-localization using per-bundle vlad.

[10] Xml for us state polygons. `http://econym.org.uk/gmap/states.xml`. Accessed: 2018-03-30.

[11] NASA Socioeconomic Data and Applications Center (SEDAC). the gridded population of the world, version 4 (gpwv4). `http://sedac.ciesin.columbia.edu/data/collection/gpw-v4/documentation`, 2017. Accessed: 2018-03-30.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.